CMD11A8 Development Board

xiom
anufacturing

™ © 1997

717 Lingco Dr., Suite 209 • Richardson, TX 75081 • (972) 994-9676 FAX (972) 994-9170
email: Gary@axman.com • web: http://www.axman.com

TABLE OF CONTENTS

TABLE OF APPENDIX

GETTING STARTED

Thank you for purchasing our CMD11A8 development system! The Axiom CMD11A8 single board computer is a fully assembled, fully functional development system for the Motorola 68HC11 Microcontroller, complete with wall plug style power supply and serial cable. To get started quickly, perform the following test now to make sure everything is working correctly:

1. Create a directory on your PC hard drive for the utility software and copy the contents of the UTL11 disk to that directory. NOTE: it is possible to run the utility software from the floppy disk but not recommended.
2. Connect one end of the supplied 9pin serial cable to a free COM port on your PC. Connect the other end of the cable to the COM port on the CMD11A8 board labeled COM1.
3. Apply power to the board by plugging in the wall plug power supply that came with the system.
4. Change to the directory containing the utility software and execute the program: **`AX11.EXE`**. Select the PC's COM port that you are using.
5. From the AX11 menu, select the option: "Terminal". This should bring up a blank terminal window.
6. Press then release the RESET button on the CMD11A8 board now.
7. If everything is working properly, you should see the Buffalo Monitor prompt in the Terminal window. Your board is now ready to use!

If you do not see the buffalo message prompt, or the text is garbage, see the **TROUBLESHOOTING** section of this manual.

SOFTWARE

There are many useful programs on the UTL11 floppy disk included with the CMD11A8 that can make developing projects on the CMD11A8 easier. You can also download the latest version of this disk free at any time from our web page at: **http://www.axman.com**.

The main programming interface to the CMD11A8 board is the AX11 program. This program communicates with the board via its COM1 port and includes a Terminal window for interfacing with other programs running on the CMD11A8, such as the Buffalo Monitor or the Basic11 interpreter.

In addition to the AX11 terminal, most communications programs will work with the CMD11A8. Even the Terminal program in Microsoft Windows™ will do an adequate job. Communications settings should be set to 9600 baud, 1start, 1stop, 8data, no parity.

See the **README** file on the utility disk for a complete listing of all programs and files available.

DEVELOPMENT PHILOSOPHY

Software development on the CMD11A8 is performed using either the Monitor or Basic utilities installed in U7 to create or assist in creating your program that is stored in RAM on U5. Your program should locate itself above the internal register block, for example $2000 (see the Memory Map section for details).

After satisfactory operation under the monitor or basic environment, your program can be written to the EEPROM in U7 by relocating it to start at $E000 then selecting "Program Code Memory" in the AX11 utility. When programming is complete your program will run automatically when the CMD11A8 is powered on or RESET is applied.

To return to Monitor or Basic development mode, run AX11 and select "Program Code Memory" then select one of the following files you wish to program:

BUF34X.S19	the Buffalo Monitor program.
BASIC11.S19	the Basic11 interpreter

If you have the 32K version of the CMD11A8 you can use U6 for external data storage or you can relocate your code to $8000 and use the extra memory for code space. If you do this, be sure to change the HC11 reset vector at $FFFE - $FFFF to point to the beginning of your code ($8000 for example).

QUICK TUTORIAL

To help reduce the time you must spend learning to develop software with the CMD11A8, we recommend you complete the following brief tutorial. This will walk you thru the complete development cycle of a very simple program. This should help you get started with the specifics of the CMD11A8 development process. Be sure to read the rest of this manual as well as the documentation on the software disk if you need further information (the buffalo manual is especially useful). Also, you should have a good reference manual for the 68HC11 Microcontroller.

For this tutorial, we will use a sample assembly program on the UTL11 software disk called HELLO.ASM. This is a simple program that just sends a text string to your PC serial port. You can substitute your own program here if you wish but, to verify everything is working properly, it's a good idea to start with something simple.

1. If you haven't done so already, verify that the CMD11A8 is connected and operating properly by following the steps under "GETTING STARTED".
2. At your PC's DOS command line prompt, change to your UTL11 software directory.
3. Execute the command: **AS11 HELLO.ASM** ↵
   This will assemble our test source code.
4. If any errors were found they would be displayed on the screen, otherwise, you should have the new file **HELLO.S19** (a Motorola hex object file) in your directory.
5. Execute the command: **AX11** ↵
   This will launch the Axiom programming interface for the CM11A8. The first time it is run you must select the PC serial port you are using. You can later change this value in the Options menu.
6. Select Terminal from the menu to see the terminal window.
7. Press and release the RESET button on the CM11A8 board. You should see the Buffalo Monitor prompt. Hit the return key ↵ to get the monitor prompt.
8. Type **LOAD T** ↵
   This will prepare buffalo to receive a program.

9. Press the Page Up key and when prompted for a file name, type: **`HELLO.S19`**
   then select [OK].
10. Make sure Echo is turned on (default) then select [OK].
    Your program will be sent to the CM11A8 board.
11. Type **`CALL 2400`** ↵
    This tells buffalo to execute the subroutine at address $2400, which is the start of our test program.
12. If everything is working properly you should see the message "Hello World" echoed back to your terminal screen then, since we return at then end of our program, a line containing the internal register status displayed by buffalo and the buffalo prompt.
13. If you do not get this message, try going thru this tutorial once more, then if still no go, see the **TROUBLESHOOTING** section in this manual

You can modify the hello program to display other strings or do anything you want. The procedures for assembling your code, uploading it to the board and executing it remain the same. Buffalo has many powerful features such as breakpoints, assembly/disassembly, memory dump and modify and program trace. Type HELP at the buffalo prompt for a listing of commands or consult the buffalo manual file on the utility disk for more information.

When you are finished with program development, you will probably want to write your program to EEPROM so that it executes automatically when you apply power to the board. The following procedure will accomplish this:

1. Use a text editor to modify HELLO.ASM. Change the start of your program, **`ROMBS`**, to **`$E000`** which is the beginning of the EEPROM in U7.
2. Remove the comment * character before the first line after **`START`**. This will initialize the stack pointer which is necessary when running outside of buffalo but should not be done while running under buffalo since buffalo must handle the stack.
3. At the DOS prompt, execute the command: **`AS11 HELLO.ASM`** ↵
   to reassemble the program. Note: you can also select "Assembler" from the AX11 main menu, which calls the batch file DO_ASM.BAT which automates this process and creates a listing file.
4. Start the programming interface: **`AX11`** ↵
5. Select "Program Code Memory" and when prompted for a file name, type: **`HELLO.S19`**
   then select [OK].
6. Follow the instructions on screen. When finished programming, remove jumpers 1,2 and 10.
7. Select "Terminal" from the menu then cycle power or press RESET on the CMD11A8 board. Your new program should start automatically.

Note: you should probably replace the return at the end of main with an endless loop or something since returning from nowhere will have unknown results when your program finishes.

To return to development mode, repeat the above steps starting with number 4, substituting BUF34X.S19 instead of HELLO.S19.

SERIAL PORTS COM1 AND COM2

COM1 is a simple, three wire asynchronous serial interface with hard wired Clear to Send (CTS) and Data Terminal Ready (DTR). It is driven by the HC11 internal SCI port using I/O pins PD0 and PD1. These two logic level signals are coupled thru a RS232 level shifter to the COM1 connector.

COM2 is a full 9-pin standard serial port with handshaking that is implemented by an R65C51 ACIA . Chapter 6 Memory Map and Appendix B have more information on the R65C51. Page three of the four page schematic found in Appendix A details the R65C51 connections.

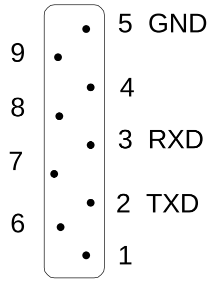
COM2 can be implemented either as an RS232 communications port or as an RS485/RS422 communications port. Option jumpers RX_SEL and 2W/4W select the interface mode between RS232, RS485, and RS422. Jumper RX_SEL selects receiver input between RS232 and RS485/RS422. Jumper 2W/4W selects transmit output between RS485 (2W) and RS422 (4W). In the RS485/RS422 mode of operation, the RTS control line is used to control transmit enable for multidrop situations. Under software control the RTS line can be enabled and disabled to provide the necessary multidrop transmitter operation.

The COM2 R65C51 ACIA receiver clock is normally derived from Y2, a 3.864 MHz crystal. An external receiver clock source can also be used by connection to the RXC pad located between C11 and C12 on the board. See Appendix B for clarification on how to implement the RXC clock source in software.

Serial Port Connections and Jumpers

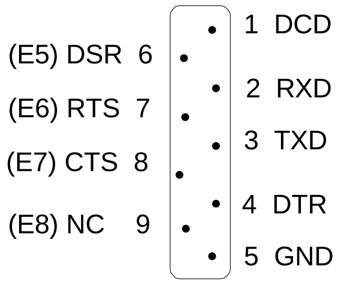
component side view

COM1 DB9S Style Connector



- Permanent jumpers between following pins:
  4 → 1 and 6
  7 → 8
- COM1 is set to connect directly to a PC serial port with a straight thru type of cable (supplied).

COM2 DB9P Style Connector



- To connect COM2 to a PC serial port you must use a NULL modem cable or NULL modem adapter.
- To change P_COM2 DB9P connector from RS232 to RS485/422 connections:
  1. Remove U13 from socket to isolate P_COM2 DB9P connector.
  2. Add jumper wire or connector from E5 thru E8 to the RS422/485 output connector.
- With U13 removed, CTS, DTR, and DCD will be idled automatically allowing the R65C51 ACIA to function normally.

COM2 Option Jumpers

RX_SEL

```
1  2  3        1 - 2  =  RS232
○  ○  ○        2 - 3  =  RS485/422
```

2W / 4W

```
1   ○          1 - 2  =  RS485 / 2W
2   ○          2 - 3  =  RS422 / 4W
3   ○
```

422/485 OUT

4 ○	A(+)	4W Transmit
3 ○	B(-)	4W Transmit
2 ○	A(+)	4W Receive, 2W Transmit / Receive
1 ○	B(-)	4W Receive, 2W Transmit / Receive

PARALLEL PORTS

The 68HC11 is configured for expanded/multiplexed mode. It uses Port B and Port C for address and data buss to external memory and memory mapped I/O devices. This leaves CPU Port D, Port A, and Port E to provide all other parallel I/O from the controller. CPU port lines are mixed as input only, output only, and some are input or output. All CPU port lines serve dual functions with internal CPU peripherals such as the timer subsystem and port A, the A/D converter on port E, and the SPI or SCI on port D.

To increase general purpose input/output capability an 82C55 peripheral port expander is memory mapped onto the data buss providing three auxiliary ports. The auxiliary port lines are all input or output configurable and are available on the AUX_PORT connector. All port lines are limited to sinking and sourcing approximately 1mA. maximum. Refer to Chapter 6 Memory Map and Appendix C for more information on the 82C55.

AUX_PORT Connector

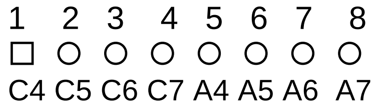
The AUX_PORT connector is a dual row 13 pin Berg-style connector (26 pins total) which is configured as follows:

```
25 23 21 19 17 15 13 11  9  7  5  3  1
 ○  ○  ○  ○  ○  ○  ○  ○  ○  ○  ○  ○  □
 ○  ○  ○  ○  ○  ○  ○  ○  ○  ○  ○  ○  ○
26 24 22 20 18 16 14 12 10  8  6  4  2
```

PIN	FUNCTION	PIN	FUNCTION	AUX_PORT ADDRESSES
1	AUX PA7	2	AUX PA6	
3	AUX PA5	4	AUX PA4	AUX PA @ B5F4
5	AUX PA3	6	AUX PA2	
7	AUX PA1	8	AUX PA0	AUX PB @ B5F5
9	AUX PC7	10	AUX PC6	
11	AUX PC5	12	AUX PC4	AUX PC @ B5F6
13	GND	14	+5	
15	AUX PC0	16	AUX PC1	AUX PORT CONTROL
17	AUX PC2	18	AUX PC3	REGISTER @ B5F7
19	AUX PB0	20	AUX PB1	
21	AUX PB2	22	AUX PB3	SEE APPENDIX C.
23	AUX PB4	24	AUX PB5	AND MEMORY MAP
25	AUX PB6	26	AUX PB7	

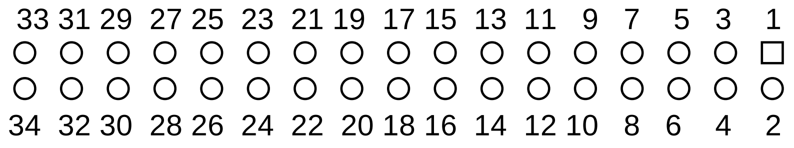
KEYPAD Connector

The Keypad Connector is an eight position connector that implements 4 bits of Aux_PortC and 4 bits of Aux_PortA as a simple keypad interface. This interface is implemented as a software keyscan. Keypad Connector pinout follows:



CPU_PORT Connector

The CPU_PORT connector is a dual row 17 pin Berg-style connector (34 pins total) which is configured as follows:



PIN	FUNCTION	PIN	FUNCTION	PIN	FUNCTION	PIN	FUNCTION
1	PA0	2	PA1	19	GND	20	+5
3	PA2	4	PA3	21	IRQ	22	XIRQ
5	PA4	6	PA5	23	VRL	24	VRL
7	PA6	8	PA7	25	VRH	26	VRH
9	+5	10	+5	27	PE7	28	PE3
11	GND	12	GND	29	PE6	30	PE2
13	PD0/RXD0	14	PD1/TXD0	31	PE5	32	PE1
15	PD2/SI	16	PD3/SO	33	PE4	34	PE0
17	PD4/SCLK	18	PD5/SEL0				

PD0 and PD1 are used by the HC11 SCI to implement COM1. PD<2:5> are used by the HC11 SPI to implement the SIMPLE_SERIAL : KEYBOARD port. These port D lines can also be used for parallel I/O, but then they will not be available for COM1 and the SIMPLE_SERIAL : KEYBOARD port. Use caution when assigning port D lines to functions other than COM1 and the SIMPLE_SERIAL : KEYBOARD port (Identified as SS : KEYBOARD on the schematic and silk screen).

KEYBOARD INTERFACE

The keyboard interface can be accomplished with software drivers through the SPI feature of the 68HC11. Such a configuration is used with a serially encoded keyboard available from the manufacturer. The keyboard port can also be used as a keypad by using PD<2:5> as column lines with E<1:4> as row lines. Note that E<1:4> are open termination's with pull-down resistors on the board. E<1:4> are ONLY connected to pins 7, 8, 9, and 10 of the SS : KEYBOARD connector. In order to use those pins of the SS : KEYBOARD connector, E<1:4> must be connected to I/O lines. AUX_PORT C is recommended because it is capable of nibble or 4 bit configuration. While such a dual, parallel matrix will work, it dedicates the SPI to keypad use only. If implemented through the SPI as a SIMPLE SERIAL port, four separate, additional SPI serial devices can be supported by the SEL lines defining E<1:4> for a total of five slave serial devices. (Note: this still requires E<1:4> be connected. Each to its own I/O line, but now we can support four additional peripheral devices.)

SS : KEYBOARD Connector

1	□	+5		
2	○	GND		
3	○	PD2/SI	------------------------	
4	○	PD3/SO		
5	○	PD4/SCLK	SPI	
6	○	PD5/SEL0	------------------------	
7	○	SEL1	------------------------	E1
8	○	SEL2	auxiliary select lines	E2
9	○	SEL3	terminated to ground (10kohm)	E3
10	○	SEL4	------------------------	E4

Note that SEL<1:4> of Axiom SS boards map directly across to E<1:4>

LCD INTERFACE

The LCD interface is connected to the data buss and memory mapped to locations B5F0 and B5F1. It supports all OPTREX™ DMC series displays up to 80 characters. Power, ground, and Vee are available at the LCD_PORT connector on the card. The potentiometer, R23, located to the right of the LCD connector is used to adjust the contrast of the LCD display by varying Vee.

LCD_PORT Connector

```
14 12 10  8  6  4  2
 ○  ○  ○  ○  ○  ○  ○
 ○  ○  ○  ○  ○  ○  □
13 11  9  7  5  3  1
```

PIN	**FUNCTION**	**PIN**	**FUNCTION**
1	GND	2	+5
3	VEE	4	A0
5	R/W	6	LCDCS
7	D0	8	D1
9	D2	10	D3
11	D4	12	D5
13	D6	14	D7

BATTERY BACKUP and RESET

Battery backup is provided only for the RF5C15 Real Time Clock. Page 4 of the schematic details the connection. The connections can be found next to the power connector J1. VBAT- is a global ground connection. VBAT+ is connected to the Vcc pin of U16 through a 1N4148 isolation diode. VBAT+ can be supplied by a standard 3Volt lithium battery. The RP5C15 requires a minimum of 2.0 volts for reliable backup. Reset is provided by an 8054 low voltage detector when Vdd falls below approximately 4.4 volts. Manual reset is provided by push button switch S1.

REAL TIME CLOCK

The RF5C15 RTC is shown on page 4 of the schematic. This device is no longer available. A RTC option available is the MK48TXX device which includes a battery RAM with real-time clock. This device is not as capable but can fit your needs for a calendar/clock reporting device.

MISCELLANEOUS OPTION JUMPERS

MODE Select Jumpers

The MODA and MODB pins of the HC11 are pulled high by two 10k resistors. This is the normal EXPANDED MODE configuration. A shunt on JP1 will take MODA to ground. A shunt on JP2 will take MODB to ground. These two jumpers allow selection of any of the following modes of operation:

JP1 - MODA	JP2 - MODB	MODE OF OPERATION
closed	closed	Special Bootstrap
closed	open	Special Test
open	closed	Normal Single Chip
open	open	Normal Expanded (default)

Buffalo Trace Jumper

The Buffalo Monitor Trace and Single Step functions can be enabled by installing option jumper **JP13**. This jumper will connect the PA3 Output Compare 5 pin to the XIRQ line for nonmaskable interrupt service for the Buffalo Trace functions. Use caution when installing this jumper that no other connections are made to the XIRQ or PA5 I/O pins of the 68HC11.

A/D Reference

The VRH and VRL lines from the HC11 are connected to +5v through R3 and to ground through R2 respectively. These two surface mount resistors are on the bottom (solder) side of the circuit board. The resistors are identified on the silk screen by their reference designators and the dashed line box that surrounds them. The appropriate resistor(s) need to be removed in order to apply an external reference to the VRH and/or VRL inputs.

EXPANSION BUS

The BUS_PORT supports off-board devices. Power (+5V), ground, address lines, data lines, and control lines are brought out to this 34 pin connector. Pin assignments are shown on page 4 of the schematic and are listed below for convenience.

	PIN	**BUS_PORT**	**PIN**	
GND	1	□ ○	2	D3
D2	3	○ ○	4	D4
D1	5	○ ○	6	D5
D0	7	○ ○	8	D6
A0	9	○ ○	10	D7
A1	11	○ ○	12	A2
A10	13	○ ○	14	A3
OE	15	○ ○	16	A4
A11	17	○ ○	18	A5
A9	19	○ ○	20	A6
A8	21	○ ○	22	A7
A12	23	○ ○	24	A13
W/R	25	○ ○	26	CS0
CS1	27	○ ○	28	CS2
CS3	29	○ ○	30	CS4
CS5	31	○ ○	32	IRQ
+5	33	○ ○	34	M2

ADDRESS DECODING AND MEMORY MAP

Address decoding is accomplished using a GAL16V8 programmable logic device. Address lines A<8:15>, AS (address strobe), R/W (read/write), and E (clock) are processed to provide the memory control signals as shown below by default. Custom configurations, differing from that shown below, are also possible. Contact the factory for assistance in redefining the memory map if required.

OE	Output enable to U5, U6, and U7
WR	Write enable to U5 direct, and to U6 and U7 through jumpers JP6 and JP10 respectively.
M1	Chip select to U5 active from 0 to 8K (with mirrored mapping) , or 0 to 32k depending on the status of JP3. See U5 JP3 selection for more information.
M2	Chip select to U6 active for the 24k between 8000 and DFFF, with the exception of B580 through B7FF inclusive. B580 to B5FF is used by P. B600 to B7FF is taken by the HC11 internal EEPROM.
M3	Chip select to U7 active for the 8k between E000 and FFFF.
P	Peripheral Access CS0 - CS7. B580 through B5FF.

All of these signals except P are active low. P is active high. Signal line M2 is also connected to the BUS_PORT expansion connector allowing M2 to work in conjunction with the CS and Address lines to implement off board, page banked memory. When M2 is used in this manner, U6 must be removed from the board.

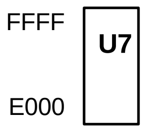
U5 is intended to be either an 8k or a 32k RAM. U6 can accommodate RAM, EEPROM, or ROM. U7 is to be used primarily for ROM but it can also accommodate EEPROM. Jumpers JP3 through JP10 determine how U5, U6, and U7 are used. The chart on the next page defines these options.

Peripheral Access 'P' is used in conjunction with A<4:6>, and AS to generate CS<0:7>. Each of these eight chip selects controls sixteen bytes in the memory map from B580 through B5FF. CS7 is used by the R65C51 UART. CS6 is used by the RF5C15 Real Time Clock. CS<5:0> are brought out to the BUS_PORT where they can be used to control peripherals external to the development board. See the Memory Map for further clarification.

Memory Selection Jumpers

The factory setting for the jumpers should be correct for the memory devices that came with your board. If you add or modify the type or size of memory, you must change the following jumpers accordingly. All jumpers are two-pin jumpers and are installed vertically.

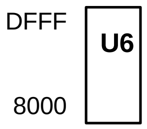
ROM or **EEPROM**



JP8	**JP9**	**JP10**	**U7 MEMORY DEVICE**
open	open	closed	8k EEPROM.
open	open	open	8k EPROM.
open	closed	open	32k EPROM.
closed	open	closed	32k EEPROM.

JP10 write protects U7 memory device when open.

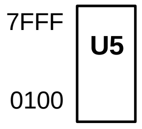
RAM or **EEPROM** or **ROM**



JP4	**JP5**	**JP7**	**JP6**	**U6 MEMORY DEVICE**
open	open	open	closed	8k RAM or EEPROM.
open	open	open	open	8k EPROM.
open	closed	closed	open	32k EPROM.
closed	open	closed	closed	32k RAM or EEPROM.

JP6 write protects U6 memory device when open.

RAM



For 32K device	**JP3 = Closed** 8K from 2000 hex to 3FFF hex and mirrored at 6000 - 7FFF. Recommended position for Buffalo Monitor and Small C operation. No segmentation occurs.
For 8K device	**JP3 = Open** 8K from 0000 hex to 1FFF hex and mirrored at 2000 - 3FFF, 4000 - 5FFF, and 6000 - 7FFF. This position will allow CPU internal Ram and I/O ports to segment the 8K address space. Use this position to run Basic11.

You can tell the size of memory devices by reading the label on top of the chip. Memory devices that contain 64 in the part number are usually 8K. Those with 256 are usually 32K.

The type of memory can also be determined by reading the chip label. If you don't recognize the memory type you can look up the part number in a catalog or device manual. If the chip is by Atmel™ or XICOR™ it is probably an EEPROM. If it has HY or SEC it is probably RAM.

Memory Map

This memory map is for an HC11A8 as used in this development board. Other HC11 devices in the A series may also be used, as well as devices in the E and F series. These optional devices differ in the amount of internal RAM, ROM, EEPROM available and the factory default value of the CONFIG register. Consult the technical reference for the specific device you are using for additional information.

FFFF FFFE	RESET Vector Address (default E000)			2 Bytes
FFFD E000	ROM or EEPROM (U7) Program			8K
DFFF B800	ROM, RAM or EEPROM (U6 or Off Board M2 Select)			10.2K
B7FF B600	HC11 Internal EEPROM			512 Bytes
B5FF		$B5FC - $B5FF	Reserved	
		$B5FB	Control Register	
	R65C51 ACIA COM2	$B5FA	Command Register	8 Bytes
		$B5F9	Status Register	
B5F8		$B5F8	Transmit / Receive Data	
B5F7		$B5F7	Control Register	
	82C55 AUX PORTS	$B5F6	Aux Port C Register	
		$B5F5	Aux Port B Register	4 Bytes
B5F4		$B5F4	Aux Port A Register	
B5F3		$B5F2 - $B5F3	Reserved	
	LCD	$B5F1	Data Register	4 Bytes
B5F0		$B5F0	Command Register	
B5EF B5E0	CS6 Real Time Clock (Refer to Appendix B, RF5C15 RTC)			16 Bytes
B5DF B5D0	CS5 TO BUS_PORT			16 Bytes
B5CF B5C0	CS4 TO BUS_PORT			16 Bytes
B5BF B5B0	CS3 TO BUS_PORT			16 Bytes
B5AF B5A0	CS2 TO BUS_PORT			16 Bytes
B59F B590	CS1 TO BUS_PORT			16 Bytes
B58F B580	CS0 TO BUS_PORT			16 Bytes

B57F		
	ROM/RAM/EPROM (U6 OR OFF BOARD M2 Select)	13.7K
8000		
7FFF		
	RAM (U5)	28.6K
1040		
	68HC11 Internal Registers. Partial listing.	
	See HC11 Reference Manual for complete listing and usage information. This register block can be relocated to any unused 4k boundary to provide a contiguous RAM space. The registers will appear as shown here on initial power up/RESET of the CMD11A1 board.	
103F	CONFIG	
103D	INIT	
103C	HPRIO	
1039	OPTION	
102F	SCDR	
102E	SCSR2	
102D	SCCR2	
102C	SCCR1	64 Bytes
102B	BAUD	
1029	SPSR	
1028	SPCR	
1026	PACTL	
1025	TFLG2	
1024	TMSK2	
1020	TCTL1	
100F	TCNT	
100E	TCNT	
100D	OC1D	
100C	OC1M	
100B	CFORC	
100A	PORTE	
1009	DDRD	
1008	PORTD	
1007	DDRC	
1005	PORTCL	
1004	PORTB	
1003	PORTC	
1002	PIOC	
1000	PORTA	
0FFF		
	RAM (U5)	3.8K
0100		
00FF		
	HC11 Internal RAM	256 Bytes
0000		

TROUBLESHOOTING

The CMD11A8 board is fully tested and operational before shipping. If it fails to function properly, inspect the board for obvious physical damage first. Ensure that all socketed IC devices are properly seated in the their sockets.

The most common problems are improperly configured communications parameters and attempting to use the wrong COM port (on the PC AND on the development board). Verify that your communications port is working by substituting a known good serial device, or by doing a loop back diagnostic. Verify that no other devices are conflicting with the port (such as a mouse, modem, etc.).

Check your hardware configuration jumpers. Make sure JP1 and JP2 are NOT jumpered. Verify the power source. You should measure 9 volts between GND and +9V test point pads on the board near J1. If no voltage is found, verify wallplug connections to 115VAC outlet and J1 power connector. Disconnect all external connections to the board except for COM1 to the PC and the wall plug. Follow these steps in the order given:

Troubleshooting Steps

1. Visual Inspection
2. Verify that mode jumpers (JP1 and JP2) are not installed
3. Verify power by checking for +9 volts between GND and +9V test point pads.
4. Verify the presence of +5 volts between the GND test point and pin 1 of the SS:KEYBOARD connector.
5. Verify U7 EPROM for proper installation (no bent pins) and proper jumper settings for the device used.
6. Re-Check the communications parameters.
7. Disconnect any peripheral devices including display, and keyboard.
8. Make sure that the RESET line is not being held low.
   Check for this by measuring across S1.
9. Verify presence of 8MHz sine wave on the crystal if possible.
10. Please check off these steps and any others you may have performed before calling so we can better help you.

Tips and Suggestions

Following are a number of tips, suggestions and answers to common questions that will solve most problems users have with the AX11 development system. This information is also available in the AX11 program under Troubleshooting. There also may be a newer version of the AX11 utility software available. You can download the latest version for free on our web page at: www.axman.com.

AX11 Program

- If you're using CMD or CMM boards and are trying to program code memory, make sure jumpers 1 and 2 are set parallel to the MC68HC11 micro, not pointing away from it.
- If you're trying to execute Buffalo, Basic, or your own code from EEPROM, make sure jumpers 1 and 2 are NOT installed.
- Be certain that the data cable you're using is bi-directional and is connected securely to both the PC and the board. Also, make sure you are using the correct serial port.
- Make sure the correct power is supplied to the board. You should only use a 9volt, 300mA adapter or power supply. If you're using a power strip, make sure it is turned on.
- If the configuration file loads (the first 100 bytes or so), but you get a time-out error when the program section begins to download, make sure the HC11 is internally configured correctly by selecting Configure Processor from the main menu. ROMON must be set to off. During program development the only bit that should be on is EEON.
- Make sure you load your code to an address space that actually exists. The 8k boards (those with part numbers ending in -8) have EPROM's beginning at address E000h. The 32k boards can be programmed starting at 8000h.
- If you are running in a multi-tasking environment (such as Windows™) close all programs in the background to be certain no serial conflict occurs.
- If programming is slow, run the batch file PAGE.BAT in the ax11 directory then see if programming is faster. If programming doesn't work following this, return to normal operation by running the batch file BYTE.BAT.
- If the Assembler or Small C compiler menu options do not work properly on your system, you can modify their operation by editing the files DO_SC.BAT (for the C compiler) and DO_ASM.BAT (for the assembler). These are the batch files that are run when their menu items are selected. Try putting a PAUSE statement at the end of the batch files. This will halt the batch file before returning to AX11 so you can see any error messages they may be giving you.
- You can reset all AX11 configuration options to their original state by deleting the file named AX11.CFG. This file will be re-created the next time you run AX11.

Code Execution

- Make sure ALL jumpers are set correctly according to your board's configuration. Read the hardware manual section on jumpers carefully if you're not sure.
- Always remember to remove jumpers 1, 2 and 10 (jumper 6 on the CMM board) after programming the code memory.
- If the board has the trace jumper, make sure it's not installed when running code from ROM or when using interrupts.
- If you programmed your code into EEPROM memory and it doesn't run, check the HC11 reset vector, located at FFFEh - FFFFh. These 2 bytes contain the address in the micro. where execution will begin when the unit is powered on. The default is E000h, which is the beginning of the 8k program address space (or the middle of the 32k program address space).

Basic11

- After programming Basic11.S19, you must remove JP1, JP2, JP10 AND JP3
- In some older versions of the Basic11 manual, the word AUTOEE was misused in place of the correct command AUTOST.

- If you want to autostart your Basic code don't forget to add a RETURN command somewhere in your programs main loop to get back to the basic interpreter. Otherwise, you'll have to remove the external EEPROM to get BASIC11 to autostart again.
- If you're getting errors or your program isn't uploading to basic11 properly from the AX11 Terminal program, try adding pacing delays. 10 or 20 (ms) should be enough.

ImageCraft C

- Your make or build should create a .MAP file. At the top of this file should be a label __START. This is where you should CALL or GO to when debugging in buffalo. Do not use your main label.

SMALLC Compiler

- If you're programming to ROM, delete the first line of the S19 record generated, since SMALL.C adds data there.
- Make sure you put the CODE, DATA, and STACK at the correct locations per your memory configuration. DATA may need to be at 0000 if you're burning a ROM, and STACK should have plenty of room.
- Don't forget to use the -R option if you're compiling for ROM.
- If Internal Registers have been re-mapped (default at 0x1000) be sure to change the #define REG_BASE accordingly.
- If you're having trouble with the shift operations (>> or <<) this is probably because the small c compiler uses a rotate thru carry instruction here. Try using inline assembly code to accomplish the shift.
- You may have trouble returning any values larger than 1 byte from functions (i.e. char functions). You will have to use global integer values instead.
- Small C does not support structures or unions.
- Small C is a free "un-supported" public domain compiler. If you're doing more than just experimenting consider purchasing a commercial quality C compiler. ImageCraft sells a good compiler for around $50, call them or Axiom Mfg. to order.

APPENDIX A

SCHEMATIC

APPENDIX B

R65C51 ACIA DATA SHEET

APPENDIX C

82C55 AUX PORTS DATA SHEET

APPENDIX D

STANDARD LCD AND KEYPAD DATA SHEETS